Implementierungsdokumentation LinkedInterval

Allgemeines

Ein `LinkedInterval` verwalten eine Menge von mehreren einfachen Intervallen vom Typ `LinkedIntervalItem` als verkettete Liste.

Im `LinkedIntervalItem` existiert ein Zeiger auf das nächste `LinkedIntervalItem` der Menge der einfachen Intervalle, sowie ein Zeiger auf die eigentlichen Daten vom Typ `SimpleInterval`.

Im `LinkedInterval` selbst existierten die Zeiger `first` und `last` die auf das erste bzw. letzte Element der Menge zeigen. Ist die Menge leer, sind beide Zeiger null. Enthält die Menge nur ein Element, zeigen `first` und `last` auf das selbe `LinkedIntervalItem`.

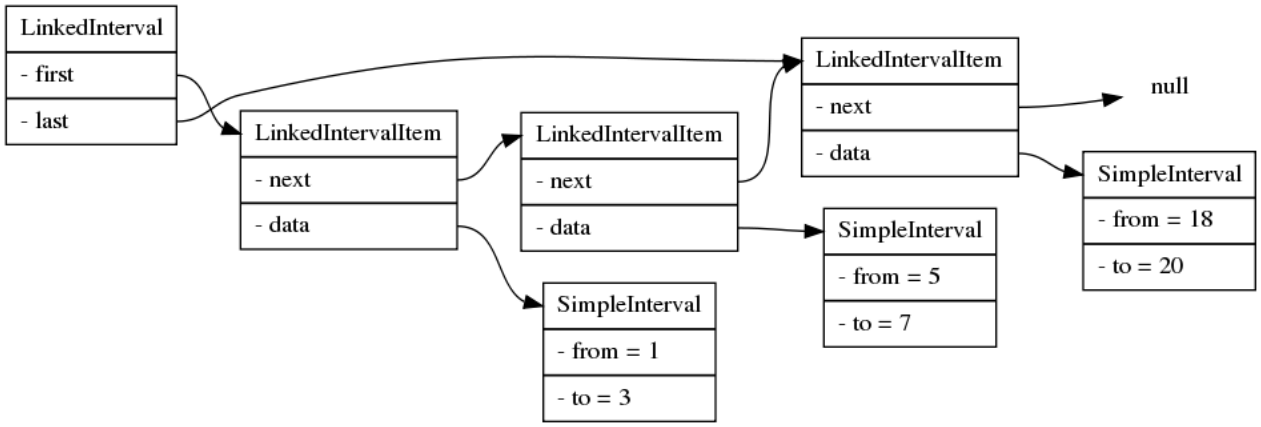

Illustration 1: Beispiel für ein LinkedInterval mit drei einfachen Intervallen von 1 bis 3, 5 bis 7 und 18 bis 20.

union()

Bei der Vereinigung von zwei Objekten des Typs `LinkedInterval` entsteht ein neues `LinkedInterval`, das die überlagerten einfachen Intervalle beider Objekte überschneidungsfrei enthält.

Da per Definition alle einfachen Intervalle einer Menge überschneidungsfrei sind und zudem noch aufsteigend sortiert sind, machen wir uns dies zu Nutze, in dem die einfachen Intervalle der beiden zu vereinigenden Intervalle beginnend beim jeweils ersten durchlaufen werden und das jeweils kleinste zu den Intervallen der Vereinigung mittels `LinkedInterval.add(SimpleInterval i);` hinzugefügt werden.

Nehmen wir als Beispiel die beiden Intervalle

`i1 := [3|5][7|9][13|15]`

und

`i2 := [1|6][8|11]`

Beim Durchlauf über alle Elemente der beiden Intervalle wird beim Vergleich folgende neue Reihenfolge der Intervalle erstellt:
`i3 := [1|6][3|5][7|9][8|11][13|15]`

Das Ergebnis wäre aber nicht korrekt, da die einfachen Intervalle nicht überlappungsfrei sind. Deswegen wird beim Einfügen eines Intervalls zur Menge in `addInterval(LinkedIntervalItem linkedIntervalItem)` überprüft, ob es eine Überlappung mit dem letzten Element der Menge gibt – ist dies der Fall, wird, statt das neue Intervall hinzu zu fügen, das letzte erweitert.

Im Beispiel passiert (vereinfacht) folgendes:

`add([1|6]);` → Leere Menge, das Intervall wird hinzugefügt

`add([3|5]);` → Innere Überschneidung mit vorigen Element, dieses Intervall wird verworfen

`add([7|9]);` → Keine Überschneidung, das Intervall wird hinzugefügt

`add([8|11]);` → Überschneidung, [7|9] wird zu [7|11] erweitert

`add([13|15]);` → Keine Überschneidung, das Intervall wird hinzugefügt.

Die resultierende Vereinigung ist

`i3 := [1|6][7|11][13|15]`

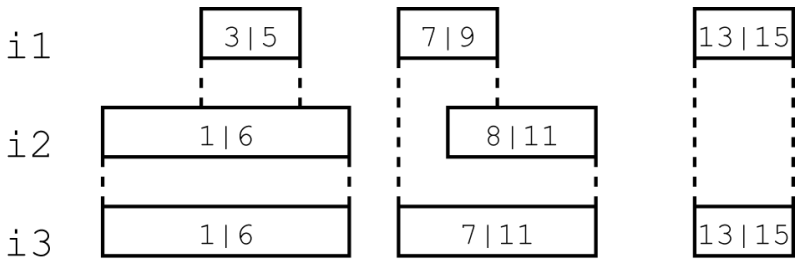

Illustration 2: Vereinigung von i1 und i2 zu i3

Schnelle Vereinigung

Eine schnelle Vereinigung ist möglich, wenn eines der beiden zu vereinigenden Intervallen leer ist, dann kann einfach das nicht-leere Intervall geklont werden.

intersect()

Bei der Überschneidung von zwei Objekten des Typs `LinkedInterval` entsteht ein neues `LinkedInterval`, das jeweils nur die Schnittmengen der einfachen Intervalle beider Objekte enthält.

Da nur diejenigen einfachen Intervalle zum Erzeugen der Überschneidung in Frage kommen, die sich mit einem oder mehreren Intervallen das jeweils anderen Objektes überschneiden, prüfen wir dies in unserer Implementierung. Hierzu wird jedes Intervall des ersten Objektes auf Überschneidung mit jedem Intervall des zweiten Objektes überprüft (mittels `LinkedIntervalItem.intersects(LinkedIntervalItem i))`, ist eine Überschneidung vorhanden wird die Überschneidung beider Intervalle zum Ergebnis hinzugefügt.

Nehmen wir als Beispiel die beiden Intervalle

`i1 := [3|5][7|9][13|15]`

und

`i2 := [1|6][8|11]`

Beim Durchlauf über alle Elemente der beiden Intervalle wird beim Prüfen auf Überschneidung festgestellt, dass sich die Elemente `[3|5]` mit `[1|6]` und `[7|9]` mit `[8|11]` überschneiden. `[13|15]` überschneidet sich mit keinem Element aus i2 und wird damit verworfen.

i3 besteht also aus der Überschneidung von `[3|5]` mit `[1|6]` und `[7|9]` mit `[8|11]`.

`i3 := [3|5][8|9]`

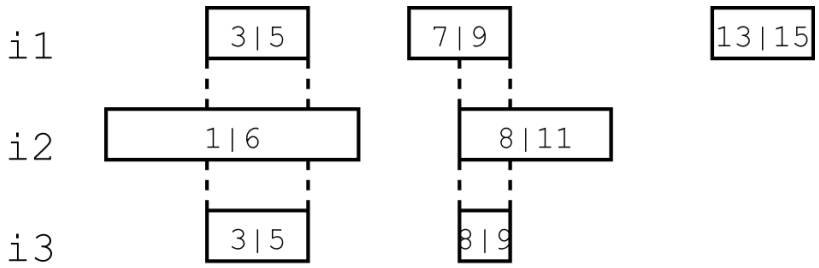

Illustration 3: Überschneidung von i1 und i2 zu i3

Schnelle Überschneidung

Eine schnelle Überschneidung lässt sich erzeugen, sobald sich keine der einfachen Intervalle der Objekte überschneiden, also wenn das erste Element des einen `LinkedInterval` hinter dem letzten Element des anderen `LinkedInterval` liegt oder das letzte Element des einen vor dem Ersten des anderen.

Trifft dies zu – die Überprüfung findet in `LinkedInterval.intersects(LinkedInterval i)` statt – ist das Ergebnis der Überschneidung eine leere Menge.